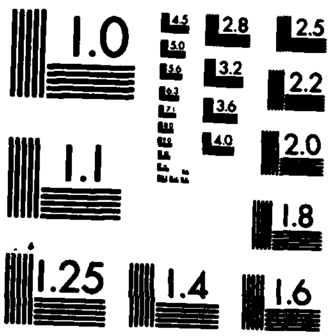
AD-A139 991 A KNOWLEDGE-BASED SYSTEM FOR LP (LINEAR PROGRAMMING) 1/1
MODELING(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA
D R DOLK 10 JAN 83 NPS-54-83-012

UNCLASSIFIED F/G 12/1 NL



NPS-54-83-012

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A KNOWLEDGE-BASED SYSTEM FOR LP MODELING

by

Daniel R. Dolk

10 January 1983

Approved for public release; distribution unlimited.

Prepared for:
Chief of Naval Research
Arlington, Va 22217

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Commodore R. H. Shumaker
Superintendent

David A. Schrady
Provost

This report was sponsored by the Foundation Research Program at the Naval Postgraduate School.

Reproduction of all of this report is authorized.

This report was prepared by:

Daniel R. Dolk

DANIEL R. DOLK, Assistant Professor
Department of Administrative Sciences

Reviewed by:

Richard S. Elster

RICHARD S. ELSTER, Chairman
Department of Administrative Sciences

Released by:

William M. Tolles

WILLIAM M. TOLLES, Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS-54-83-012	2. GOVT ACCESSION NO. A139991	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Knowledge-Based System for LP Modeling		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Daniel R. Dolk		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N; RR000-01-10 N0001483WR30104
11. CONTROLLING OFFICE NAME AND ADDRESS Chief of Naval Research Arlington, Va 22217		12. REPORT DATE 10 January 1983
		13. NUMBER OF PAGES 22
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Model Management
Knowledge-Based Systems
Linear Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Mathematical programming software has tended to be more algorithm-oriented and less concerned with model management capabilities. This report studies the objectives, functions, and subsequent structures required of a generalized model management system. A prototype software package implementing some of these structures is described.

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A KNOWLEDGE-BASED SYSTEM FOR LP MODELING

Daniel R. Dolk
Naval Postgraduate School

1. INTRODUCTION

The focus of this paper is on linear programming (LP) software in the context of model management and decision support. As a result, we will not be interested in the algorithmic properties of LP or math programming (MP) codes, but rather their applicability to more generalized modeling environments wherein models can be linked or decomposed in a manner which frees the user from having to know the internal representation which the algorithms require. In particular, we will look at the objectives and functions of a generalized model management system and how these require a knowledge-based modeling capability. We will then describe a partial implementation of a knowledge-based modeling system for LP models called the Generalized eXperimental Math Programming system (GXMP).

2. LP AND MODEL MANAGEMENT

Most, if not all, LP software has operated on the assumption that users know enough about their model to select the correct package to use to obtain a computer solution. In the decision support environment, the user, in fact, may not be aware of what models are needed or available to solve the problem at hand. In this case, it becomes the task of the model management system to orchestrate the appropriate models and data to solve a user-described problem (Figure 1).

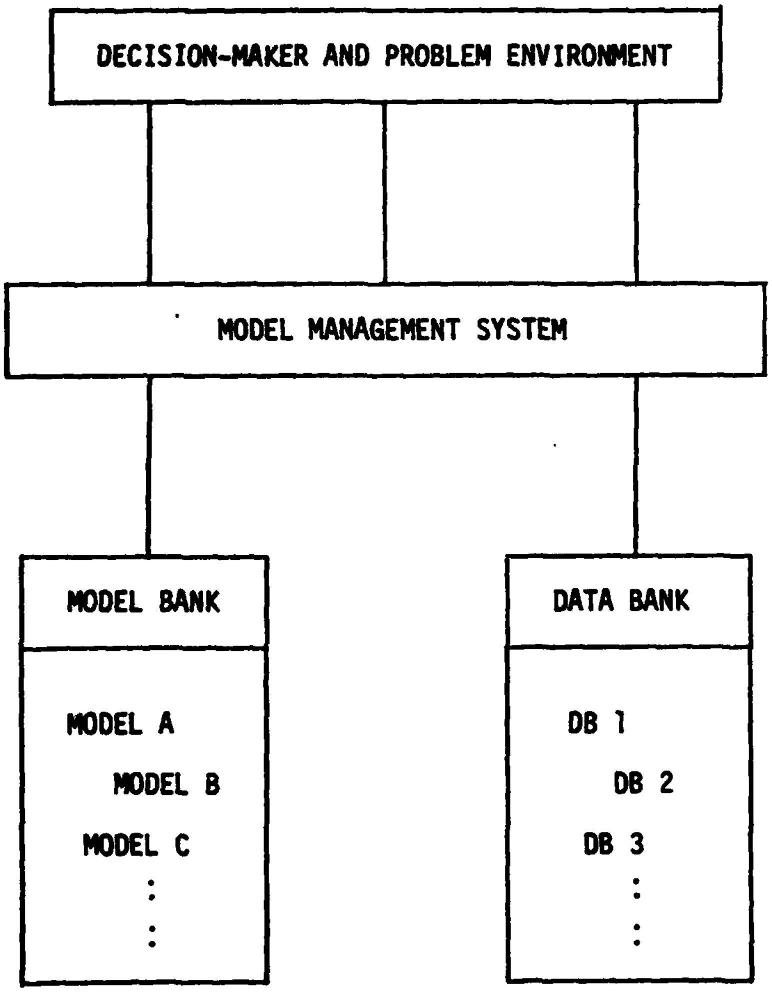


Figure 1: Role of Model Management in Decision Support

In order for an MMS to function in this capacity, it clearly requires a knowledge-based capability for representing knowledge about models. The following kinds of knowledge should be contained in the MMS, for example:

1. Which situations particular models are applicable to;
2. Model integrity statements;
3. Information required to instantiate a model;
4. Which solution algorithm(s) to employ to solve a model;
5. How to link models to other models;
6. How to decompose models into submodels;
7. Interpretation of model results.

In addition to knowledge representation and storage, an MMS must also have one or more generalized problem-solving or inferencing capabilities for manipulating the knowledge. This "reasoning" capability should encompass heuristic as well as deterministic reasoning so that, for example, the MMS can pattern-match user-supplied problem descriptions with available model and data instances. The first order predicate calculus has been suggested as one means of representation (Bonczek et al, 1981) which is well-suited for deterministic kinds of inferences. Semantic networks have been considered to support more heuristic kinds of reasoning within a modeling environment (Elam et al, 1980). The GXMP system is based on the concept of knowledge abstractions, or in the case of modeling, model abstractions (Dolk, 1982; Konsynski and Dolk, 1982), a knowledge representation scheme which attempts to combine the strengths of deterministic and heuristic inferencing (Dolk, 1982).

A model abstraction consists of three parts: data objects, procedures acting upon those objects, and assertions or facts (knowledge) concerning the relationships between data objects and procedures (Figure 2). For MP models, data objects correspond to constraint equations and model parameters, procedures

DATA OBJECTS

(Ex: Constraint Equations,
Index Sets)

PROCEDURES

(Ex: Add-Constraint-Model,
Modify-Index-Set-Value,
Linear (Equation))

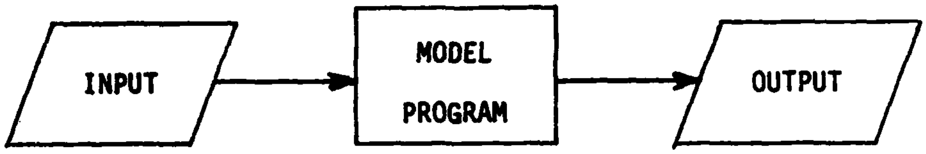
ASSERTIONS

(Ex: Linear (Constraint),
Linear (Obj-Func))

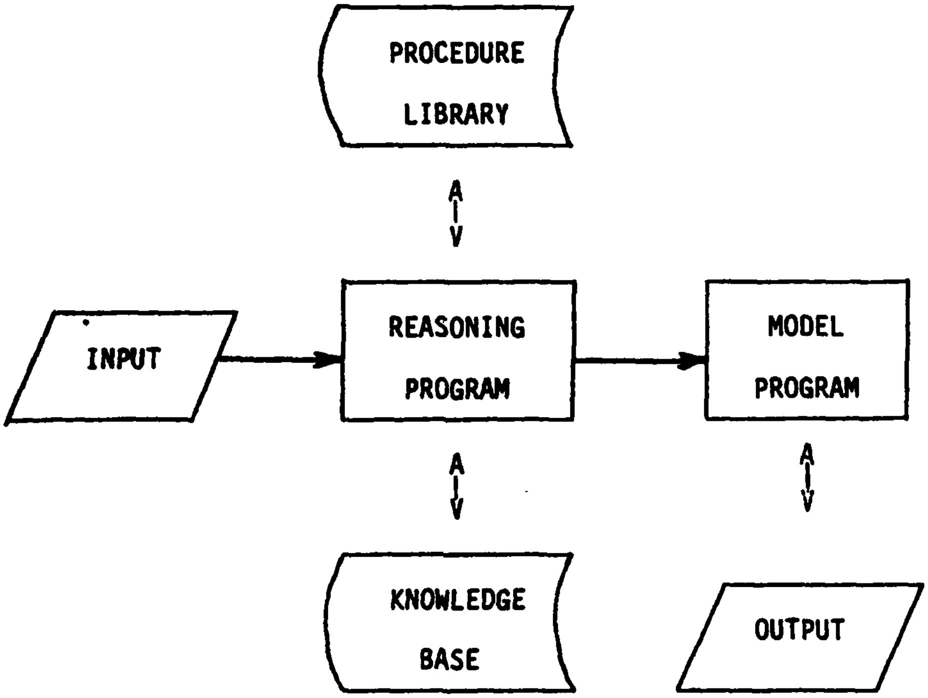
Figure 2. Model Abstraction Structure

are operators which act upon the data objects (ADD-CONSTRAINT-TO-MODEL, e.g.), and assertions specify integrity conditions ("constraints must be linear in an LP model," e.g.).

A modeling system with a knowledge-based capability alters the traditional view of modeling software (Figure 3). Typically, math programming software has been algorithmically oriented wherein the input data are prepared by the user-modeler and fed into a "black box" program which applies the algorithm to supply the results. Knowledge about the models is supplied externally by the modeler in preparing the input and internally by the programmer in specifying the algorithm. *In neither case is this knowledge* accessible to other users who see only the input and the "black box." In the knowledge-based modeling system, however, this structure is expanded to abstract the knowledge pertinent to a model into a separate data component of the system. Furthermore, the algorithms



(a) Traditional Modeling Software System



(b) Knowledge-Based Modeling System

Figure 3. Difference Between Traditional and Knowledge-Based Modeling Systems

are stored as data in a procedure library to be used by the reasoning module in constructing a model program to solve the problem described by the user. For those familiar with rule-based expert systems, the knowledge base would consist of rules expressed in a *IF* <antecedents(s)> *THEN* <consequent(s)> fashion and the reasoning program would correspond to a rule interpreter (Duda and Gaschnig, 1981).

3. THE GXMP SYSTEM

The remainder of this paper describes a "first pass" implementation of a knowledge-based model management system for LP models called GXMP. Much of the effort to date on GXMP has been involved with establishing a suitably powerful modeling language for expressing MP models. As a result, the knowledge-based capabilities of the system are still very limited. Development phases for incorporating these features into GXMP are presented in the concluding section.

Figure 4 shows the salient implementation features of GXMP. The system is written in FORTRAN on a Vax-11/780 using the XMP linear programming library (Marsten, 1980) and the ADBMS database management system (Hershey and Messink, 1975). The overall goal of the GXMP system is to serve as a user-friendly virtual machine for LP (and eventually MP) algorithms. Thus, although the XMP library was chosen as the source of the simplex algorithm, there is no restriction on using other algorithms providing they accept input in matrix form.

The components of GXMP are shown in Figure 5. There are five different database types:

1. The directory catalogs all instances of models, data objects, abstractions, and databases currently in the system;

2. The abstraction database stores model abstractions and serves as the knowledge base of the system;

3. The procedure database stores the subroutines from the XMP library including the source code;

4. *The equation database* stores an instance of a model expressed in the GXMP modeling language;

5. The parameter database stores the associated numerical parameters of a model instance (coefficient and right-hand-side values, e.g.).

Interface with the user occurs at two levels: a menu dialog for selecting the system functions to be performed and a modeling language for expressing LP

Model Management System for LP

FORTRAN

XMP Library (Marsten, 1980)

ADBMS (Hershey and Messink, 1975)

Knowledge-Based (model abstractions)

User-Friendly Virtual Machine for LP Algorithms

Figure 4. Features of GXMP Implementation

DATABASES

- Directory (catalog)
- Abstraction (knowledge base)
- Procedure (XMP library)
- Equation (expressed in modeling language)
- Parameter (coefs, rhs, etc.)

MENU DIALOG

MODELING LANGUAGE

SOLUTION REPORTER

Figure 5. Components of the GXMP System

models in a mathematical fashion. The menu dialog accommodates the "naive" user while the modeling language is geared toward the modeler. The solution reporter generates LP solutions in terms of the variables which the user specifies during model creation. Examples of these various aspects of the system are shown in the sample problem of the next section.

The GXMP modeling language is designed to allow modelers to represent expressions in quasi-mathematical notation. It is somewhat similar to FORTRAN and contains several key features:

1. User-supplied variable names in place of mathematical variables (self-documenting):

Math-Var	Description	GXMP-Name
i	city	CITY
j	city	CITY
X_{ij}	shipment from city i to city j	SHIPMENT (i,j)
D_j	demand from city j	DEMAND (j)

2. Use of inequality signs for expressing constraints.

3. MIN, MAX, SUM operators.

4. Use of mathematical indexes in equations.

5. FOR clause for specifying conditions on indexes and instantiating indexes.

   Ex: To express $\sum_{i=1}^{n} X_{ij} = D_j$ for $j = 1,\ldots,n;\ \ j<>i$

```
SUM(i)[SHIPMENT(i)] = DEMAND(j)
FOR  i  OVER CITY,  j  OVER CITY,  i<>j  $
```

6. Use of index expressions

```
Ex:  CONSUMPTION(p,t) + INVENTORY(p,t) -
        INVENTORY(p,t-1) = SUPPLY(p,t)
        FOR  p  OVER COMMODITY,  t  OVER PERIOD  $
```

Several other syntactical features and restrictions should be mentioned:

1. All indexes must be expressed in lower case and cannot exceed four characters in length. Indexes "max" and "min" are reserved names (see 4. below).

2. Simple index expressions of the form <index> <"+" or "-"> <integer or index> are allowed, e.g.: X(i-1,t+1).

3. Indexes must be used consistently over all equations and each index appearing in an equation must be identified in that equation. The first restriction states that index i, for example, must be the same

in each equation in which it appears. Furthermore, different index names cannot be used in different equations to represent the same index (unless the index forms a link). The second restriction requires identifying an index whenever it appears. This leads to a certain amount of redundant specification but results in fully documented equations.

4. Each instance of SUM must be followed by one or more indexes enclosed in parentheses and the summand enclosed in square brackets, e.g.:

is expressed as: $$\sum_{i,j} X_{ij}$$

```
SUM (i,j)[X(i,j)]
```

5. Conditions on indexes can be expressed in conjunction with index identification, thus we can express:

```
0 <= X(i,j) <= DEMAND(i,j) FOR i OVER CITY,
                           j OVER CITY, i <> j $
```

to indicate that the constraint holds for all i,j except $i = j$. ("<>" indicates the "not equal" operator.) The reserved words "max" and "min" can be used to refer to the highest and lowest cardinal values of an index respectively, thus

$$\sum_{i,j}^{n-1} X_{ij}$$

can be expressed as:

```
SUM(i,j)[X(i,j)] FOR i<max, j<max, i OVER...
```

6. All decision variables must be grouped onto one side of a constraint. Thus if C and F are decision variables:

```
C(p,t) + F(p,t) = S(p,t) + F(p,t-1) FOR ...
```

will not be translated properly and should be rewritten as:

```
C(p,t) + F(p,t) - F(p,t-1) = S(p,t) FOR ...
```

7. Each unique combination of decision variable and indexes may appear only once in a particular equation. Thus the following objective function with decision variable DV:

```
MAX(SUM(i,j)[REV(i,j)*DV(i,j)-EXPNS(i,j)*DV(i,j)])..
```

will not be translated properly and should be rewritten as:

```
MAX(SUM(i,j)[(REV(i,j)-EXPNS(i,j))*DV(i,j)])..
```

Note this restriction does not obviate a decision variable from appearing more than once in an equation as long as it has different index expressions each time (see sample in 6. above).

8. Only constraint and objective function expressions are allowed. No intermediate expressions may be inserted. This means that although it would be convenient to write the objective function above as:

```
UNIT-PROFIT(i,j) = REV(i,j) - EXPNS(i,j)
PROFIT = SUM(i,j)[UNIT-PROFIT(i,j)*DV(i,j)]...
MAX(PROFIT)
```

   it is not currently permitted in the GXMP.

Only the last point is a serious restriction of the language in any sense, yet it does not pose any conceptual difficulty involving implementation. It was imposed primarily to expedite the development time of a "first pass" version of the system.

An example of the language applied to a sample problem is presented in the next section.

4. A GXMP EXAMPLE

This section presents a simple problem to show how the GXMP operates in creating and solving a model. The model we are going to use is the school rezoning problem discussed in Hillier and Lieberman (1974, pp. 196-201). The gist of the problem is to minimize the distance students need tc be bused to school while maintaining a specified racial balance in each school. A mathematical representation is displayed in Figure 6.

GXMP recognizes three different kinds of parameters:

1. index sets (i and j in this model);
2. decision variables (X);
3. numerical parameters (B, W, S, T);

Index sets correspond to the indexes in the mathematical representation but their values in the GXMP are always character quantities rather than numerical

Variable	Description
X_{ij}	No. of students in tract i assigned to school j
D_{ij}	Distance from tract i to school j
B_i	No. of black students in tract i
W_i	No. of white students in tract i
S_j	Capacity of school j
T	Parameter such that $.5-T \leq$ racial balance $\leq .5+T$

$$\min \sum_i \sum_j D_{ij}X_{ij} \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

$$\text{st } \sum_j X_{ij} = B_i + W_i \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

(all students are assigned to schools)

$$\sum_i X_{ij} \leq S_j \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

(school capacity not exceeded)

$$\sum_i (.5-T-W_i/(B_i+W_i))X_{ij} \leq 0 \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

$$\sum_i (.5-T-B_i/(B_i+W_i))X_{ij} \leq 0 \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

(racial balance)

$$X_{ij} \geq 0 \qquad \text{for } i=1,\ldots,10;\ j=1,2,3$$

(nonnegativity)

Figure 6. Mathematical Description of School Rezoning Model

quantities as in the mathematical case. This significantly increases the documentation value of the system. Decision variables are the quantities being solved for by the simplex method. Numerical parameters are those quantities with numerical values, usually coefficients or right-hand-sides in the equations.

When entering parameters, always enter the index sets first since subsequent parameters will depend upon them. This is done in our example by specifying TRACT for index i and SCHOOL for index j. For each parameter entered, the system will prompt the user for parameter type, data type (not prompted for index sets and decision variables), documentation, indexes (if any) that the parameter depends upon, and data values to be entered at this time (if any).

When entering parameters which are indexed, include the indexes as part of the name selected ("DISTANCE(i,j)" for "X" for example). The indexes must be specified in lower case and cannot exceed four characters in length. They must also be used consistently across parameters and equations, i.e. "i" cannot refer to TRACT in one case and SCHOOL in another. Neither can "i" refer to TRACT in one case and "j" refer to TRACT in another (unless TRACT forms the basis of a network problem which is not the case here). The best way to avoid confusion is to construct a table that correlates the variables in the mathematical description with their GXMP counterparts (Figure 7). Once the desired parameters have been added to the model, a listing of those parameters can be obtained (Figure 8).

The next step in specifying the model is to enter the objective function and the constraints using the modeling language described in the previous section. The important rules to remember when entering equations are:

1. No intermediate expressions are allowed, thus there will be only one objective function expression;

Model Variable	Parameter Type	GXMP Name
i	Index set	TRACT
j	Index set	SCHOOL
X_{ij}	Decision var	STUDENTS(i,j)
D_{ij}	Parameter	DISTANCE(i,j)
B_i	Parameter	BLACKS(i)
W_i	Parameter	WHITES(i)
S_j	Parameter	SCHOOL-CAPAC(j)
T	Parameter	THETA

Figure 7. GXMP Parameter Names for School Rezoning Model Variables

2. All parameters appearing in an equation must be defined in the parameter database prior to equation entry;

3. Each separate index appearing in an equation must be instantiated in the FOR clause and, like parameter entry, indexes must be used consistently across equations;

4. Summation is expressed by

   SUM(index1,index2,...)[summand];

5. Always end each equation with a "$".

Equations are classified as either constraints (CONSTR) or objective function (OBJFCN) and assigned an id number as well. If the user does not specify an id number when entering an equation, the system will assign one automatically. Once the equations have been entered, they can be displayed in a manner similar to that of the parameters (Figure 9).

```
*Type   <CR>  for  Previous Menu
           1  for  Equation Operations
           2  for  Parameter Operations
           3  to   Solve Model

*2

*Type   <CR>  for  Previous Menu
           1  to   Add Parameter(s)
           2  to   Delete Parameter(s)
           3  to   Modify Parameter(s)
           4  to   Specify Values for Parameters
           5  to   Link Parameters
           6  to   Display Parameters in Model
           7  to   Display Parameter Values

*6

*Parameter name (20 chars max)
 (If adding, include indexes if any, eg:  DEMAND(i,j)
 (Type ALL for all, Hit <CR> to end parameter entry):

*ALL
                            Data
Parameter-Name      Type    Type    Description

BLACKS(i)           PA      R       Black students in
     i: TRACT                       tract i
DISTANCE(i,j)       PA      R       Distance from tract i
     i: TRACT                       to school j
     j: SCHOOL
SCHOOL              IS      C       Schools
SCHOOL-CAPAC(j)     PA      R       School j student capacity
     j: SCHOOL
STUDENTS(i,j)       DV      R       Students in tract i
     i: TRACT                       assigned to school j
     j: SCHOOL
THETA               PA      R       Parametric quantity
TRACT               IS      C       School tracts
WHITES(i)           PA      R       White students in tract i
     i: TRACT
```

Figure 8. GXMP Session Displaying School Rezoning Model Parameters

```
*Type <CR>  for  Previous Menu
         1  for  Equation Operations
         2  for  Parameter Operations
         3  to   Solve Model
*1

*Type <CR>  for  Previous Menu
         1  to   Add Equation(s) to Model
         2  to   Delete Equation(s) from Model
         3  to   Renumber Equation(s) in Model
         4  to   Display Equation(s) in Model
         5  to   List Variables in Equations
         6  to   Compile Equations
*4

*Type <CR>  to   Return to Previous Menu
         1  if   Equation is a Constraint
         2  if   Equation is Objective Function
*2

*Type equation id (0 for all):
#0

OBJFCN-Id     Equation

    10        MIN(SUM(i,j)[DISTANCE(i,j)*STUDENTS(i,j)])
                        FOR i OVER TRACT, j OVER SCHOOL  $

     1 OBJFCN equations in model

*Type <CR>  to   Return to Previous Menu
         1  if   Equation is a Constraint
         2  if   Equation is Objective Function
*1

*Type equation id (0 for all):
#0

CONSTR-Id     Equation

    10        SUM(j)[STUDENTS(i,j)] = BLACKS(i) + WHITES(i)
                        FOR i OVER TRACT, j OVER SCHOOL  $

    20        SUM(i)[STUDENTS(i,j)] = SCHOOL-CAPAC(j)
                        FOR i OVER TRACT, j OVER SCHOOL  $

    30        SUM(i)[(.5 - THETA - WHITES(i)/(WHITES(i)
                        BLACKS(i)))*STUDENTS(i,j)] <= 0.
                        FOR i OVER TRACT, j OVER SCHOOL  $

    40        SUM(i)[(.5 - THETA - BLACKS(i)/(WHITES(i)
                        BLACKS(i)))*STUDENTS(i,j)] <= 0.
                        FOR i OVER TRACT, j OVER SCHOOL  $

    50        STUDENTS(i,j) >= 0.
                        FOR i OVER TRACT, j OVER SCHOOL  $

     5 CONSTR equations in model
```

Figure 9. GXMP Session Displaying School Rezoning Model Equations

The primary strength of the GXMP modeling language is the indexing capability which allows equations to be specified symbolically and concisely. Note that each constraint equation in the school rezoning problem in Figure 6 actually represents multiple constraint instances. Constraint 10, for example, symbolizes i constraint instances. This economy of expression due to the mathematical nature of the modeling language allows users to represent large models with very few equations.

Furthermore, the language is robust in that it enforces a high degree of independence between the equations and the data. It is sufficient to observe that tracts and schools could be added or dropped from the parameter database for the school rezoning problem without having to make a single change to the equations in Figure 9. Similarly, equations could be added or deleted without altering the parameter data values. This would not be the case if a language required that each constraint be enumerated explicitly. Clearly there is a connection between model independence and the power of a modeling language.

Once a model has been fully specified (parameters, parameter values, and equations), it is ready to be solved. The model solution process takes place as shown in Figure 10. The appropriate model abstraction instance for the model is located and a SOLVE predicate located within the abstraction. The SOLVE predicate lists which XMP routines need to be invoked and the order of invocation in order to effect a solution for the model. In a batch environment, a job stream is established consisting of JCL commands, the appropriate XMP routines retrieved from the procedure library, and parameter data and equations retrieved from the parameter and equation databases. The job stream is submitted as a batch job whose result file is then processed interactively in a later session. In a totally interactive setting, the XMP routines are executed dynamically and the solution generated online. The obvious disadvantage of the

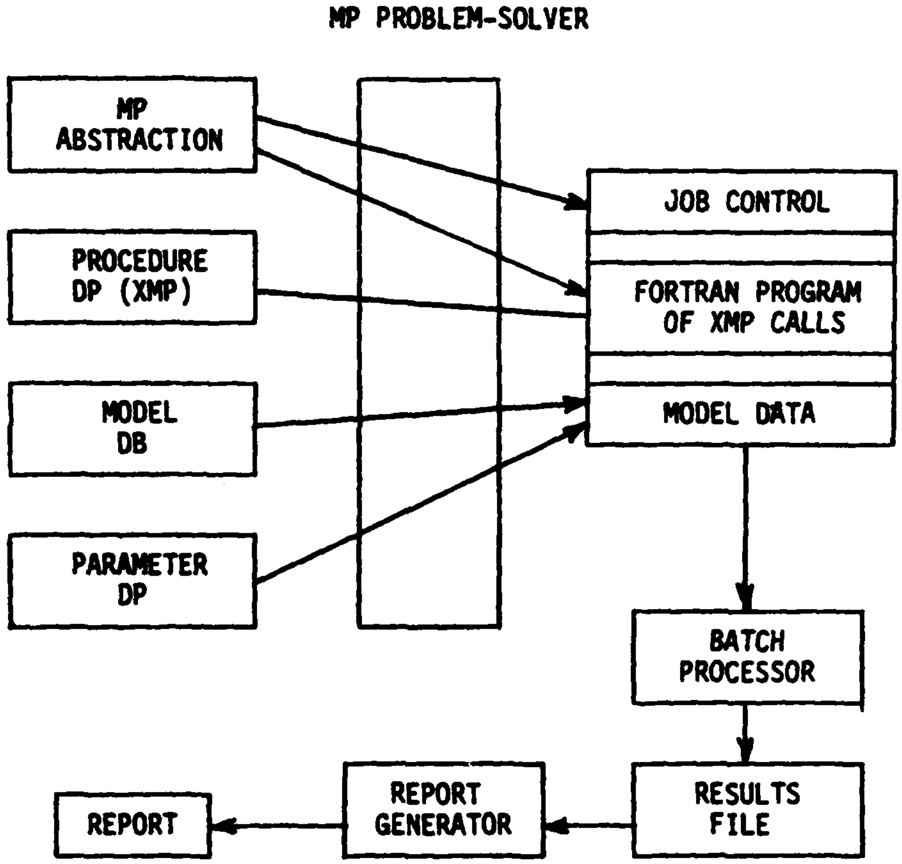


Figure 10. The GXMP Model Solution Process for Batch Jobs

latter approach is that online processing may be too time-consuming for all but small scale LP problems. Because our example is small, the interactive approach is used in this case.

Once the model has been solved via the XMP routines, the solution reporter is invoked to report the results (Figure 11). The reporting of linear programming results has been a long-standing problem especially with large models having many variables. At the root of the problem is the naming convention that systems impose on the unfortunate user which often results in confusion as to which variable is which in the report. Names like "Variable 1"

* * * * * GXMP SOLUTION REPORTER * * * * *

Model Type : LP-EON
Model Instance: SCHOOL-REZONING

Number of Linear constraints : 19
Number of Variables (incl. slacks & artificials): 49

Variable Identification	Value
STUDENTS(TRACT1,JEFFERSON)	0.45000000D+03
STUDENTS(TRACT2,JEFFERSON)	0.40000000D+03
STUDENTS(TRACT3,JEFFERSON)	0.50000000D+03
STUDENTS(TRACT4,WASHINGTON)	0.50000000D+03
STUDENTS(TRACT5,WASHINGTON)	0.40000000D+03
STUDENTS(TRACT6,WASHINGTON)	0.45000000D+03
STUDENTS(TRACT7,JEFFERSON)	0.15000000D+03
STUDENTS(TRACT7,WASHINGTON)	0.30000000D+03
STUDENTS(TRACT8,HAMILTON)	0.50000000D+03
STUDENTS(TRACT9,WASHINGTON)	0.50000000D+02
STUDENTS(TRACT9,HAMILTON)	0.35000000D+03
STUDENTS(TRACT10,HAMILTON)	0.45000000D+03

Remainder of variables are slacks/artificials

Constraint for Which Variable is Basic	Value
SCHOOL-CAPAC(HAMILTON)	0.30000000D+03
WHITE-RATIO(JEFFERSON)	0.55583337D+03
WHITE-RATIO(WASHINGTON)	0.91668144D+00
WHITE-RATIO(HAMILTON)	0.41075002D+03
TRACT-ASSIGNMENT(TRACT6)	0.89166689D+02
TRACT-ASSIGNMENT(TRACT9)	0.74008336D+03
TRACT-ASSIGNMENT(TRACT8)	0.14825001D+03

Value of Objective Function: -0.49650000D+04

Constraint	Dual Variable
TRACT-ASSIGNMENT(TRACT1)	-0.14000000D+01
TRACT-ASSIGNMENT(TRACT2)	-0.27999998D+01
TRACT-ASSIGNMENT(TRACT3)	-0.89999992D+00
TRACT-ASSIGNMENT(TRACT4)	-0.13000000D+01
TRACT-ASSIGNMENT(TRACT5)	-0.40000001D+00
TRACT-ASSIGNMENT(TRACT6)	-0.60000002D+00
TRACT-ASSIGNMENT(TRACT7)	-0.14000000D+01
TRACT-ASSIGNMENT(TRACT8)	-0.17000001D+01
TRACT-ASSIGNMENT(TRACT9)	-0.12000000D+01
TRACT-ASSIGNMENT(TRACT10)	-0.15000001D+01
SCHOOL-CAPAC(JEFFERSON)	0.19999993D+00
SCHOOL-CAPAC(WASHINGTON)	0.00000000D+00
SCHOOL-CAPAC(HAMILTON)	0.50000006D+00
WHITE-RATIO(JEFFERSON)	0.00000000D+00
WHITE-RATIO(WASHINGTON)	0.00000000D+00
WHITE-RATIO(HAMILTON)	0.00000000D+00
BLACK-RATIO(JEFFERSON)	0.00000000D+00
BLACK-RATIO(WASHINGTON)	0.00000000D+00
BLACK-RATIO(HAMILTON)	0.00000000D+00

Figure 11. GXMP Display of School Rezoning Model Solution

or "Demand(3,4)" are just not descriptive enough in most cases. Furthermore, users are often required to enumerate each name as input which can be extremely tedious for large problems. Imagine having to input "DEMAND(i,j)" for i and j equal to 100.

The GXMP subverts this problem largely through the power of its modeling language. The user need only enter each parameter name in the model once, generically as it were. Thus, "DEMAND(i,j)" is the only GXMP entry necessary for a parameter representing the demand from city j for a product at city i. After the model with this parameter is formulated and solved, the solution (assuming DEMAND(i,j) is a decision variable) is displayed with index values substituted for i and j, e.g.:

```
DEMAND(DALLAS,SEATTLE)        1124.
DEMAND(DALLAS,LA)             3962.
DEMAND(TUCSON,SEATTLE)         836.
          .                     .
          .                     .
```

The decision variables are fully documented at the back end with this convention while requiring a minimum of input at the front end. The user supplies only the names (15 characters or less) for each "generic" variable. The same convention is employed in prompting the user for parameter data values once the parameter names have been defined. Thus, usage is consistent throughout the process from model formulation to model solution.

Another user-friendly feature of this approach is the ability to focus on a subset of the solution. This can be done in the solution reporter by specifying explicitly which variables the user wants to see. Thus, one can specify STUDENTS(TRACT5,WASHINGTON) to see only that value, or more useful, specify STUDENTS(*,WASHINGTON) to see the number of students from each tract who are to be bused to the Washington school. The * "wild card" character

provides a powerful feature for looking at only the parts of the solution which a user is particularly interested in. The advantages of this approach over wading through a massive printout to find a limited set of solution variables are readily apparent, especially for large models with many decision variables.

Notice that in addition to fully identifying the decision variables in the solution basis, GXMP also provides an equivalent capability for the dual variables. This is accomplished by assigning each equation an equation name in addition to a unique numeric identification. (Note: GXMP has not yet been modified to display these names which is why they do not appear in Figure 9.) Thus equation CONSTR 10 is TRACT-ASSIGNMENT, CONSTR 20 is SCHOOL-CAPAC and so forth. Thus the dual variables can be associated with their corresponding equation appropriately indexed as in Figure 11. This feature helps document the equations and improves the readability of the solution report.

5. CONCLUSIONS AND FURTHER RESEARCH

The development of the GXMP system has been scheduled in three phases. This paper has discussed the results of the first phase which is the design and implementation of a model management system for LP models. The current version of GXMP is essentially a user-friendly top end for LP software with built-in capabilities for model management. Further work is necessary to make fuller use of these features.

The second phase is concerned with extending the knowledge-based nature of the system. Although the apparatus for a knowledge-based modeling system is already in place, many theoretical issues remain to be studied before implementation can occur. In particular, the following areas need to be resolved:

1. What kinds of knowledge to represent about models.
2. How to represent this knowledge (the model abstraction needs to be defined more carefully).

3. What kind of inferencing and pattern-matching technique(s) to use with regard to models.

The third phase of GXMP development is to extend the system to a generalized model management system (GMMS). This involves incorporating other classes of models (e.g.: simultaneous equation estimation, discrete event simulation, etc.) into the knowledge-based framework. This will test the versatility of the model abstraction concept and may entail extensions to the modeling language and compiler. Another important issue to be addressed in this stage is the development of a technique for linking models to form composite models as well as the inverse operation of decomposing models into simpler components which can then be relinked after solution.

REFERENCES

1. Bonczek, Robert B., Holsapple, Clyde W. and Whinston, Andrew B. A generalized decision support system using predicate calculus and network data base management, Operations Research, 29, 2 (1981), 263-281.

2. Dolk, Daniel R. The uses of abstraction in model management, Ph.D. Dissertation (unpublished), The University of Arizona, 1982.

3. Duda, R. O. and Gaschnig, J. G. Knowledge-based expert systems come of age, Byte, September 1981, pp. 238-282.

4. Elam, Joyce J., Henderson, John C. and Miller, L. W. Model management systems: An approach to decision support in complex organizations, Technical Report No. 80-08-04, Department of Decision Sciences, University of Pennsylvania, 1980.

5. Hershey, Ernest A. and Messink, Paul W. A data base management system for PSA based on DBTG 71, ISDOS Working Paper No. 88, University of Michigan, 1975.

6. Konsynski, B. and Dolk, D. Knowledge abstractions in model management, Proceedings of DSS-82, June 1982.

7. Marsten, Roy E. The design of the XMP linear programming library, TOMS, 7, 4 (1981), 481-497.

DISTRIBUTION LIST

	No. Copies
Office of Research Administration Code 0121 Naval Postgraduate School Monterey, CA 93943	1
Knox Library Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Defense Technical Information Center Cameron Station Alexandria, VA 23314	2
Professor Gordon H. Bradley Code 52Bz Naval Postgraduate School Monterey, CA 93943	1
Professor Gerald G. Brown Code 55Bw Naval Postgraduate School Monterey, CA 93943	1
Professor Daniel R. Dolk Code 54Dk Naval Postgraduate School Monterey, CA 93943	5
Professor Richard S. Elster, Chairman Code 54Ea Naval Postgraduate School Monterey, CA 93943	1
Professor Norman Lyons Code 54Lb Naval Postgraduate School Monterey, CA 93943	1
Professor Norman F. Schneidewind Code 54Ss Naval Postgraduate School Monterey, CA 93943	1
Professor Roger Weissinger-Baylon Code 54Ws Naval Postgraduate School Monterey, CA 93943	1

	No. Copies
Computer Center Library Code 0141 Naval Postgraduate School Monterey, CA 93943	1

END

FILMED

5-84

DTIC